MAC TR-113

ON REDUCIBILITY AMONG COMBINATORIAL PROBLEMS

Paul Peter Herrmann

December 1973

This research was supported by the National
Science Foundation under research grant GJ-34671

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

CAMBRIDGE MASSACHUSETTS 02139

ON REDUCIBILITY AMONG COMBINATORIAL PROBLEMS

Abstract: A large class of combinatorial problems have been shown by Cook and Karp to be computationally equivalent to within a polynomial. We exhibit some new problems in this class, and provide simpler proofs for some of the known reductions.

CHAPTER 1: THE PROBLEM. TERMINOLOGY.

(1.1) The Problem.

(For the terminology see (1.3).)

In 1970 S.A. Cook stated the problem whether a deterministic T.M. (Turing machine) can do in polynomial time what a nondeterministic one can do in polynomial time. His conjecture which has gained support since then is that the answer is negative. He showed that every language recognition problem (i.e. every decision problem) solvable in polynomial time by a nondeterministic T.M (we call this class of problems $\mathbb{NP}$) can be reduced in polynomial time to the satisfiability problem (i.e. whether for a given propositional formula there is a truth assignment making it true) and also to a few other problems in $\mathbb{NP}$ ([1]). We call such problems polynomial-complete in $\mathbb{NP}$ or, for short, p-complete.

In 1972, R.M. Karp listed 21 decision problems and showed that they are p-complete ([7]). In the meantime the family of known p-complete problems has been extended by S. Sahni (CS Dept., Cornell Univ.) and others.

Cook's problem has become an important issue in complexity theory ([10]). Its solution would provide some important clues about how to approach a large class of notoriously intractable computational problems in mathematical programming, operations research, theorem proving, and related areas.

(1.2) Comment.

A general characterization of p-complete problems that allows one to decide algorithmically whether a combinatorial decision problem is p-complete or solvable in polynomial time w.r.t. (with respect to) the input lenght would be very desirable. There are deceptive relations between combinatorial problems. For example, a family $\{S_i\}$, $i \in \{1,2,\dots,m\}$, of finite subsets of some domain has a system of distinct representatives iff for every finite k and any choice of k distinct indices $i_1, i_2, \dots i_k$, the subsets $S_{i_1}, S_{i_2}, \dots, S_{i_k}$ contain between them at least k distinct elements. One might be tempted to guess that the problem can only be decided by an exhaustive search. But Marshall Hall discovered an algorithm for its solution that takes only polynomial time ([6]).

We started this work looking for some characterization of p-complete problems as mentioned above. The goal has not been achieved. In Chapter 2 and 3 we are going to report on the by-products of our efforts. Both chapters have introductory sections.

(1.3) Terminology.

We assume the reader is familiar with the papers by Cook and Karp mentioned in (1.1).

We are considering decision problems and always assume that they are given in some encoded form over a finite alphabet. So they become language recognition problems.

Let P be a decision problem and L be some representation (encoding) of P over a finite alphabet Σ. We have $L \subset \Sigma^*$ +). Every element of L

+) Σ^* denotes the set of all finite strings that can be formed by elements of Σ, including the empty string.

represents an instance of P. - Assume P' is a special case (a subproblem) of P, that means using the same encoding as for P, P' is represented by a language L' and $L' \subset L$. Then we call P' a <u>contraction</u> of P. - Analogously, if P" is a generalisation of P, that means using the same encoding as for P, P" is represented by a language L" and $L'' \supset L$, then we call P" an <u>extension</u> of P.

The class of problems solvable in polynomial time by a nondeterministic T.M. is denoted by $\mathbb{NP}$, the class of problems solvable in polynomial time by a deterministic T.M. by $\mathbb{P}$. The time is always considered to be a function of the input length. We are only dealing with one-tape T.M.'s.

Given a problem P_1 as a language L_1 over a finite alphabet Σ_1, and a problem P_2 as a language L_2 over a finite alphabet Σ_2. Assume there is a mapping $f: \Sigma_1^* \to \Sigma_2^*$ s.t. for every $w \in \Sigma_1^*$, $f(w) \in L_2$ iff $w \in L_1$. f is called a <u>reduction</u> from P_1 to P_2. If there is a deterministic T.M. that performs the mapping f in polynomial time, then we say P_1 is Karp-reducible, or <u>K-reducible</u> to P_2. For that we use the notation

$$P_1 \xrightarrow{K} P_2 .$$

Some standard encodings of the problems involved are implicit in our use of this notation.

Cook's notion of reducibility is different. Given two decision problems P and Q. Assume we have an oracle for solving Q in time 1. Then, according to Cook, P is reducible to Q in polynomial time iff there is an algorithm which decides P and may use the oracle for Q (more than once) and runs in polynomial time. Then we say, P is Cook-reducible, or <u>C-reducible</u> to Q, and we use the notation

$$P \xrightarrow{C} Q .$$

Obviously, when P $\overset{K}{\longrightarrow}$ Q, then P $\overset{C}{\longrightarrow}$ Q. But it is not known whether C-reducibility implies K-reducibility. Elsewhere, when we write 'P is reducible to Q ', we mean K-reducibility.

If s is a finite string of literals, we denote the length of s, i.e. the number of literals in s, by $\nu(s)$.

Assume a decision problem P_1 is K-reducible to P_2. Assume P_i is given as a language L_i over a finite alphabet Σ_i, i = 1, 2. Assume the reduction f: $\Sigma_1^* \to \Sigma_2^*$ is performed by a T.M. M. Let $\tau(w)$ be the time M takes to compute f(w), and let $\sigma(w)$ be $\nu(f(w))$. We define

$$t(n) = \max \{ \tau(w): \nu(w) \leq n \}$$

and call t(n) the <u>reduction time</u> of f (w.r.t. M).

We define

$$s(n) = \max \{ \sigma(w): \nu(w) \leq n \}$$

and call s(n) the <u>reduction space</u> of f (w.r.t. M). - Observe that s(n) is not the <u>work space</u> needed by M to perform f.

Whenever we refer to time or space of a reduction we mean the reduction time or the reduction space, resp., as defined above.

If t(n) is linear (quadratic), we say f is time-linear (time-quadratic). If s(n) is linear (quadratic), we say f is space-linear (space-quadratic).

Assume the reduction $P_1 \overset{K}{\longrightarrow} P_2$ is such that f restricted to L_1 is a bijective mapping onto L_2, and f^{-1} gives a reduction $P_2 \overset{K}{\longrightarrow} P_1$. Then we call P_1 <u>isomorphic</u> to P_2 w.r.t. f.

Assume there is a problem P $\in \mathbb{NP}$ s.t. for every Q $\in \mathbb{NP}$ there is a reduction Q $\overset{K}{\longrightarrow}$ P. Then we call P <u>K-p-complete</u> in $\mathbb{NP}$, or sometimes just <u>p-complete</u>. - If there is a problem P' $\in \mathbb{NP}$ s.t. for every Q $\in \mathbb{NP}$ there is a reduction Q $\overset{C}{\longrightarrow}$ P', we call P' <u>C-p-complete</u> (in $\mathbb{NP}$).

We are going to use Karp's names for the p-complete problems he listed in [7], except we call SUBSET SUM what he calls KNAPSACK. These names will be written in capitals. - A p-complete contraction of SAT is the one where we confine ourselves to propositional formulas in conjunctive form. We call this problem CF-SAT. - CF-SAT with at most three literals per clause or exactly three literals per clause is also p-complete. We refer to both cases by 3-CF-SAT and specify in the context which one we mean.

A network is a graph with weighted arcs, where the weights are non-negative integers. The network may have sources with specified maximum (integer) supplies and sinks with specified minimum (integer) demands.

(1.4) Remark.

Though other parameters are sometimes used, it is common to give the computational complexity (in time and space) of a problem as a function of the input length $\nu(w)$, where w is the input, i.e. an encoding of the given instance of the problem. Hence, this computational complexity depends on the encoding and thus on the alphabet Σ chosen. However, it turns out that within a range of encodings for a given problem, encodings that we informally would like to call 'standard encodings' (cf. (3.2)), the computational complexity is only slightly affected when the encoding is being changed. If the computational complexity is of some polynomial degree d, a change of the encoding may lead to a polynomial degree of 2d or 3d. But a change of the encoding will not result in converting a complexity that is of higher than polynomial degree into a complexity of polynomial degree, and vice versa. In this sense our different encodings are invariant. The only exception to this remark is the difference between unary (or tally) and radix notation for integers which may differ exponentially. We shall always assume integers are represented in radix notation, say base two for definiteness.

We doubt that the length parameter ν is always the most suitable one for expressing the computational complexity of the problems with which we are dealing. Probably, a parameter or a vector of parameters closer to the logical structure of our decision problems would be more revealing in particular cases. But length will be our main parameter in order to maintain comparability among the several differently structured problems we consider. We do not know a better one for this purpose.

Do this for every clause and form the union of all the unions (i). Let B denote this union. Then, A is satisfiable iff B is $\frac{1}{3}$-satisfiable.

(2.2) NWF with two flow intervals.

We are going to consider network flows (NWF). Always assume there is one source s and one sink t in the network. All pertinent data are supposed to be integer valued.

Assume for a flow f_i along Arc i we have the constraints $0 \leq f_i \leq a_i$, where a_i is a nonnegative integer. There is an algorithm by Ford-Fulkerson-Edmonds to determine the maximal feasible flow in the network. The computational complexity of the algorithm is $O(n^3)$, where n is the number of nodes ([3]). This algorithm starts with a feasible flow, e.g. the zero-flow, and augments it step by step.

Now allow upper and lower bounds for the flows f_i, s.t.

$$(0 \leq) \; b_i \leq f_i \leq a_i$$

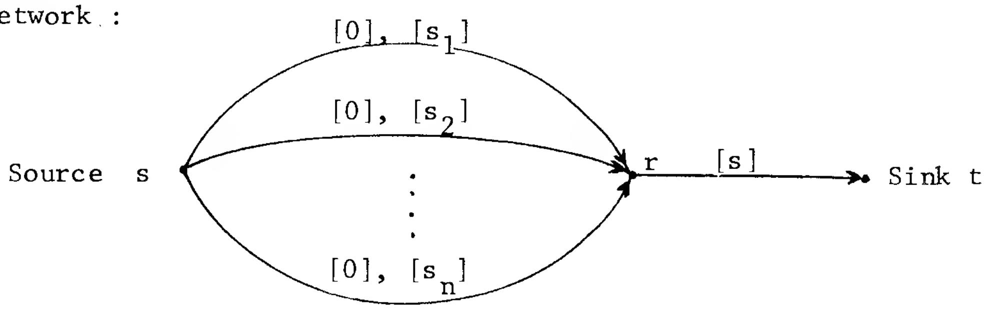
for every arc i. Here the first question is whether there is a feasible flow at all. But one can start with the zero flow and apply the algorithm mentioned before repeatedly, namely once for increasing a flow f_i along Arc i, given $f_i < b_i$, in order to get $b_i \leq f_i$ ($\leq a_i$), and, at worst, once for every i. If the computation becomes stymied no feasible flow exists. The overall computation is, at worst, $O(n^7)$. The procedure was outlined by Lawler ([8]).

To get a p-complete problem we allow two flow intervals for an arc i such that either $d_i \leq f_i \leq c_i$ or $b_i \leq f_i \leq a_i$ for given a_i, b_i, c_i, d_i; $a_i \geq b_i \geq c_i \geq d_i$. The decisive question is whether there is a feasible flow. This problem is called <u>NWF with two flow intervals</u>.

The problem is obviously in $\mathbb{NP}$. We show

SUBSET SUM $\longleftrightarrow$ NWF with two flow intervals .

Given a set of positive integers $S = \{ s_1, s_2, \ldots , s_n \}$, and a positive integer s . The Subset Sum Problem : Is there a subset S' of S such that $\sum_{s_i \in S'} s_i = s$? - For the reduction we use the following network :



There are n arcs from s to r the ith of which allows a flow of value 0 or s_i. From r to t the flow has to have value s. The Subset Sum Problem has a solution iff there is a feasible flow from s to t in the network.

<u>(2.3)</u> CTP with quadratic optimization function.

Given a network with sources and sinks. A source may have a limited supply, and at each sink a demand has to be satisfied. An arc must carry a flow between 0 and some positive upper bound, the capacity of the arc. There is a cost c_i per unit associated with the flow f_i along the directed arc i.

The problem whether there is a flow of size v with a cost not exceeding k, for given v and k, is a wellknown linear programming problem, often called the Capacitated Transshipment Problem (<u>CTP</u>). (We assume that all the pertinent data are integer valued.) If there is a solution, then there is also an integer solution ([5]). The optimization function is

$$\sum_{i \in A} f_i c_i ,$$

where A is the set of arcs of the network.

Remark: Lawler gives an algorithm for CTP with a computational complexity $O(v \cdot n^2)$, where n is the number of nodes in the network ([8]). But since we assume that the data including v are given in radix notation, the algorithm is not polynomial-time bounded with respect to the input length. We do not know whether CTP is in $\mathbb{P}$.

In CTP, an arc i that carries a flow of value f_i contributes $c_i f_i$ to the cost. Now we allow this cost contribution to be quadratic in f_i:

$$c_{2i} \cdot f_i^2 + c_{1i} \cdot f_i + c_{0i} ,$$

where c_{2i}, c_{1i} and c_{0i} are the coefficients that determine the cost contribution of arc i. So the optimization function becomes quadratic in the f_i's.

The problem whether there is a flow v of cost zero (for given v) is K-p-complete. We call this problem CTP with quadratic optimization function. - (We are considering integer data and integer solutions only.)

To prove this we show the reduction

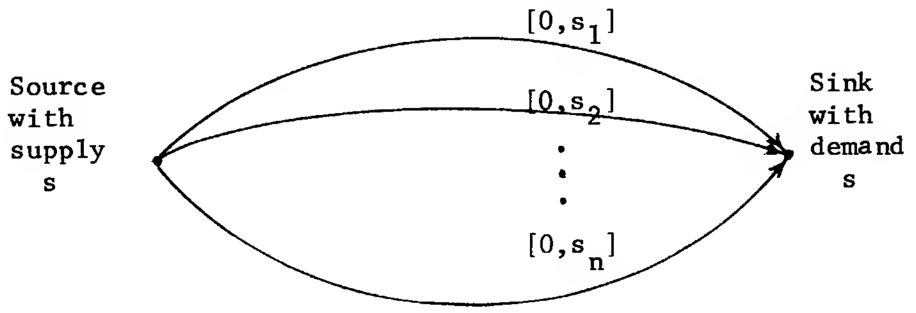
$$\text{SUBSET SUM} \xrightarrow{K} \text{CTP with quadratic optimization function} .$$

The problem is obviously in $\mathbb{NP}$.- Given a set of positive integers

$$S = \{ s_1, s_2, \ldots, s_n \},$$

and a positive integer s. The Subset Sum Problem: Is there a subset S' of S, such that $\sum_{s_i \in S'} s_i = s$?

For the reduction we construct the following network:



There are n arcs between the source and the sink. Along Arc i the flow value must be in the interval $[0, s_i]$, $i = 1, 2, \ldots, n$.

Consider the cost function

$$\sum_{i=1}^{n} f_i \, (s_i - f_i) \; .$$

The Subset Sum Problem has a solution iff there is a feasible flow of size s with cost zero.

(2.4) Summary.

Theorem 1: The following problems are K-p-complete in $\mathbb{NP}$:

a) $\frac{1}{3}$-DF-SAT ,

b) NWF with two flow intervals ,

c) CTP with quadratic optimization function .

The definitions and proofs are given in (2.1), (2.2), and (2.3).

(2.5) 3-NODE COVER is p-complete.

In the Node Cover Problem (NODE COVER) we have a graph G and an arbitrarily given positive integer k. The question is whether all the arcs of G can be covered by at most k nodes (whether there is a node cover of size k).

Now assume k is a function of the number of nodes n. This is a different problem that we denote by NODE COVER(k(n)). Here the decisive question is whether there is a node cover of size $\lceil k(n) \rceil$. For example, NODE COVER($\frac{n}{2}$) is the problem that asks whether there is a node cover of size $\lceil \frac{n}{2} \rceil$. (We let n always denote the number of nodes in the graph.)

x_1 for the first x, x_2 for the second x, x_3 for the third x, $\bar{x}_4$ for the first $\bar{x}$, and $\bar{x}_5$ for the second $\bar{x}$. - Extend F by

$$\wedge\ (\bar{x} \vee x_1) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3) \wedge (\bar{x}_3 \vee x_4) \wedge (\bar{x}_4 \vee x_5) \wedge (\bar{x}_5 \vee x),$$

that means

$$\wedge\ (x \to x_1 \to x_2 \to x_3 \to x_4 \to x_5 \to x) \ .$$

Do this for all variables occuring in F to get F'. Make sure that a new variable is introduced for every literal of F.

Example:

$$F = (x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee z) \ .$$

$$\begin{aligned} F' = {} & (x_1 \vee y_1 \vee \bar{z}_1) \wedge (\bar{x}_2 \vee \bar{y}_2 \vee z_2) \wedge (x_3 \vee \bar{y}_3 \vee z_3) \\ & \wedge (\bar{x} \vee x_1) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3) \wedge (\bar{x}_3 \vee x) \\ & \wedge (\bar{y} \vee y_1) \wedge (\bar{y}_1 \vee y_2) \wedge (\bar{y}_2 \vee y_3) \wedge (\bar{y}_3 \vee y) \\ & \wedge (\bar{z} \vee z_1) \wedge (\bar{z}_1 \vee z_2) \wedge (\bar{z}_2 \vee z_3) \wedge (\bar{z}_3 \vee z) \quad . \end{aligned}$$

Assume F has 3p literals and m different propositional variables. Then, F' has

$$3p + (2 \cdot 3p + 2m) = 9p + 2m$$

literals each of which is incompatible with at most three others. Hence, if G' is the corresponding graph of F', we have $G' \in \mathfrak{G}_3$.

Obviously, F is satisfiable iff F' is satisfiable. F' is satisfiable iff G' has a node cover of size $2p + (3p + m)$. This follows immediately from the structure of F' and from the following reductions as given by Karp ([7]):

$$\text{CF-SAT} \ \xrightarrow{K} \ \text{CLIQUE} \ \xrightarrow{K} \ \text{NODE COVER} \ .$$

<u>(2.6)</u> NODE COVER ($\mathfrak{m}$) is p-complete.

We refer to the definitions given in (2.5).

From Karp's reductions

$$\text{CF-SAT} \overset{K}{\longrightarrow} \text{CLIQUE} \overset{K}{\longrightarrow} \text{NODE COVER}$$

it follows immediately that NODE COVER (2n/3) is p-complete. The following theorem is a generalisation thereof.

<u>Theorem 3</u>: For any fixed rational $\alpha \in (0, 1)$,

NODE COVER (αn) is K-p-complete in $\mathcal{NP}$.

<u>Proof</u> of Theorem 3: We are going to show

$$\text{3-CF-SAT} \overset{K}{\longrightarrow} \text{NODE COVER } (\alpha n) .$$

Let F be a propositional formula in conjunctive form with exactly three literals per clause. Assume F has p clauses. Let G be the corresponding graph of F. G has $n = 3p$ nodes.

We know from Karp's reductions that F is satisfiable iff G has a node cover of size $2n/3$.

First, assume $\alpha \in (0, \frac{2}{3})$. We would like to reduce NODE COVER (2n/3) to NODE COVER (αn). For that, add q_1 isolated nodes to G; call the new graph G_1. Then, G_1 has $3p + q_1$ nodes. There is a node cover of 2p nodes for G iff there is a node cover of 2p nodes for G_1. We want

$$\lceil \alpha \cdot (3p + q_1) \rceil = 2p \text{ , or}$$
$$\lceil \alpha \cdot (n + q_1) \rceil = \frac{2}{3}n .$$

According to this equation, q_1 is essentially a linear function of n.

Now assume $\alpha \in (\frac{2}{3}, 1)$. G has $n = 3p$ nodes. Add a complete graph of q_2 nodes to G; call the extended graph G_2. G has a node cover of 2p nodes iff G_2 has a node cover of $2p + q_2 - 1$ nodes. We want

$$\lceil \alpha \cdot (3p + q_2) \rceil = 2p + q_2 - 1 \text{ , or}$$
$$\lceil \alpha \cdot (n + q_2) \rceil = \frac{2}{3}n + q_2 - 1 .$$

Obviously, q_2 is essentially a linear function of n .

<u>(2.7)</u> NODE COVER ($\sqrt[m]{n}$) is p-complete.

With respect to Theorem 3 one may ask for further generalisations. Our next theorem gives one answer.

<u>Theorem 4</u>: Given a fixed integer m greater than 1.

a) NODE COVER ($\sqrt[m]{n}$) is K-p-complete.

b) NODE COVER ($n - \sqrt[m]{n}$) is K-p-complete.

<u>Proof:</u> For example, we can reduce NODE COVER (n/2) to the problems given in a) and b). Then the proof is analogous to the proof of Theorem 3, i.e. we add isolated nodes or complete graphs to the given graph to get the desired ratio between n and the size of the node cover.

To carry out the reductions, we start with a graph G which has n nodes. Assume n is even (otherwise add an isolated node).

In Case a) we add q_1 isolated nodes to G and call the new graph G_1. G_1 has $n + q_1$ nodes. G has a node cover of size n/2 iff G_1 has a node cover of size n/2. Thus we want

$$\frac{n}{2} = \sqrt[m]{n + q_1} \; .$$

In Case b) we add a complete graph with q_2 nodes to G and call the new graph G_2. G_2 has $n + q_2$ nodes. G has a node cover of size n/2 iff G_2 has a node cover of size $\frac{n}{2} + q_2 - 1$. Thus we want

$$\frac{n}{2} + q_2 - 1 = n + q_2 - \sqrt[m]{n + q_2} \; , \text{ or}$$
$$\frac{n}{2} + 1 = \sqrt[m]{n + q_2} \; .$$

<u>Remark:</u> Since both q_1 and q_2 are $O(n^m)$, the reduction space itself is $O(n^m)$. We do not know better reductions and conjecture that the reduction space for any reduction

$$\text{NODE COVER (n/2)} \xrightarrow{K} \text{NODE COVER (} \sqrt[m]{n} \text{)}$$

is $O(n^m)$.

the problem is whether there is a subset of k columns that covers all rows. - One could formulate dual problems in the same way for SET PACKING and SET COVERING.

(2.9) 3-CHROMATIC NUMBER is p-complete.

The problem 3-CHROMATIC NUMBER asks whether a given graph is three-colorable. It was shown by Larry Stockmeyer that this problem and the contraction restricting it to planar graphs are p-complete ([12]).

CHAPTER 3: ON REDUCTIONS AMONG K-p-COMPLETE PROBLEMS.

(3.0) In this chapter we report some efforts made to investigate the computational complexity of the reductions among p-complete problems. A motivation is given in (3.1). In Sections (3.2) and (3.3) we give a few observations on encoding and space complexity. The rest of the chapter exhibits a few new reductions we found among known p-complete problems.

Space complexity of reductions

(3.1) By Cook's Reduction Theorem ([1]) every problem solvable in nondeterministic polynomial time is K-reducible in polynomial time to SAT. The reduction time (i.e. the polynomial) depends on the problem, i.e. on the special nondeterministic T.M. and thus also on the encoding.

Let Π_r be a problem having deterministic time lower bound n^r. Assume, for example, there is a reduction $\Pi_r \overset{K}{\longrightarrow}$ SAT that has reduction time $t_r(n)$ and reduction space $s_r(n)$. t_r and s_r are polynomials in n. Assume SAT has (deterministic) time upper bound $\tau(n)$. Then,

$$t_r(n) + \tau(s_r(n))$$

is an upper bound on the deterministic time complexity of Π_r.

If degree $t_r(n) < r$, then $\tau(s_r(n))$ must be at least $O(n^r)$. Hence, τ must be at least $O(n^{r/\deg(s_r)})$. This observation may allow us to prove large polynomial lower bounds on complete problems such as SAT even if we are unable to prove that SAT $\notin \mathbb{P}$.

Therefore we were looking for problems with deterministic polynomial time lower bounds of high degree that can be reduced to some p-complete problem, e.g. to SAT, easily, at least such that it takes space bounded by a lower degree polynomial. We did not achieve what we wanted but became interested in the computational complexity of the reductions.

(3.2) Time and space complexity of a reduction depends on encodings of problems as well as on the reduction device. Here we use one-tape T.M.'s.

The objects of the known p-complete problems are propositional formulas, graphs, graphs with weighted arcs, finite sets of integer equations, or sets of subsets of a finite domain. For each of these problems we have two or three encodings that we would like to call standard encodings. We are going to outline what kind of encodings we have in mind.

Assume we have in each case some convenient finite alphabet Σ. When we distinguish items of the same type, like variables or nodes, we use indexing and binary notation. For example, we could represent five nodes of a graph by N0, N1, N10, N11, N100, where $N \in \Sigma$.
Integers are represented in binary notation.

In case of propositional formulas we think about the following encodings:

clause-by-clause, where essentially every clause is a list of the literals occuring in it;

the incidence matrix given row by row, consisting of 0's and 1's, one row for each clause and one double column for each variable (a double column because the variable may occur affirmative or negative);

an encoding like the previous one, but 0's, +1's and -1's in the matrix and a single column for each variable, using +1 for an affirmative and -1 for a negative occurence of a variable.

In case of graphs we think about the encodings:

a list of all nodes followed by a list of all arcs, an arc given as a pair of nodes;

the incidence matrix given row by row using 0's and 1's, rows for the nodes and columns for the arcs;

the adjacency matrix given row by row.

In case of subsets of a finite domain we think about the encodings:

a list of all subsets, where every subset is a list of the elements occuring in it;

the incidence matrix given row by row, a row for every subset and a column for every variable in the domain.

For a set of equations we think about the encodings:

equation-by-equation;

the coefficient matrix of the system of equations, given row by row.

These are essentially the encodings we refer to when we use the term '_standard encoding_'.

(3.3) The reduction space of the reductions within the family of p-complete problems was considered in some detail. One can find standard

p-complete problems given in some standard encoding, v is not linear (on a one-tape T.M.), but in many cases quadratic in the input length n, sometimes even worse but not exceeding $O(n^3)$.

Some new reductions among known p-complete problems

(3.4) 3-CF-SAT $\longleftrightarrow$ EXACT COVER

Given a propositional formula F in conjunctive form with exactly three literals per clause. Assume F has m clauses and r variables occur (uncomplemented or complemented). Let C_i be the clauses, i = 1, 2, ..., m , and x_j be the variables, j = 1, 2, ..., r. Form the following array:

	x_1	$\overline{x}_1$	x_2	$\overline{x}_2$	x_3	$\overline{x}_3$		x_r	$\overline{x}_r$	
C_1	X			X	X					X
C_2		X				X				X
C_3	X				X			X		X
.										.
.										.
.										.
C_m						X				X
	X		X		X			X		

There is a row for every clause C_i and a double column for every pair $(x_j, \overline{x}_j)$. A cross X occurs in Row C_i and Column x_j (or $\overline{x}_j$) iff in Clause i the literal x_j (or $\overline{x}_j$) occurs. In addition, we have one cross for every row in an additional column at the right hand side, and one cross for every double column at the bottom of the array.

The domain of the Exact Cover Problem is the set S of all these crosses, i.e. each cross is an element in S. We form a system of subsets

$3m \leq 2r$. This shows that our reduction w.r.t. the encodings outlined above is space-linear.

2. A space-linear reduction SAT $\xrightarrow{K}$ 3-CF-SAT was given by Fischer, Meyer and Paterson ([4]). Hence we have a space-linear reduction SAT $\xrightarrow{K}$ EXACT COVER.

3. It was shown by Michael J. Fischer that our reduction 3-CF-SAT $\xrightarrow{K}$ EXACT COVER can be modified to a reduction CF-SAT $\xrightarrow{K}$ EXACT COVER that is quadratic in space.

<u>(3.5)</u> CF-SAT $\xrightarrow{K}$ directed HC [+]).

Karp ([7]) gives a reduction NODE COVER $\xrightarrow{K}$ directed HC that is rather complicated and quadratic in space. The reduction we are going to describe here is linear in space.

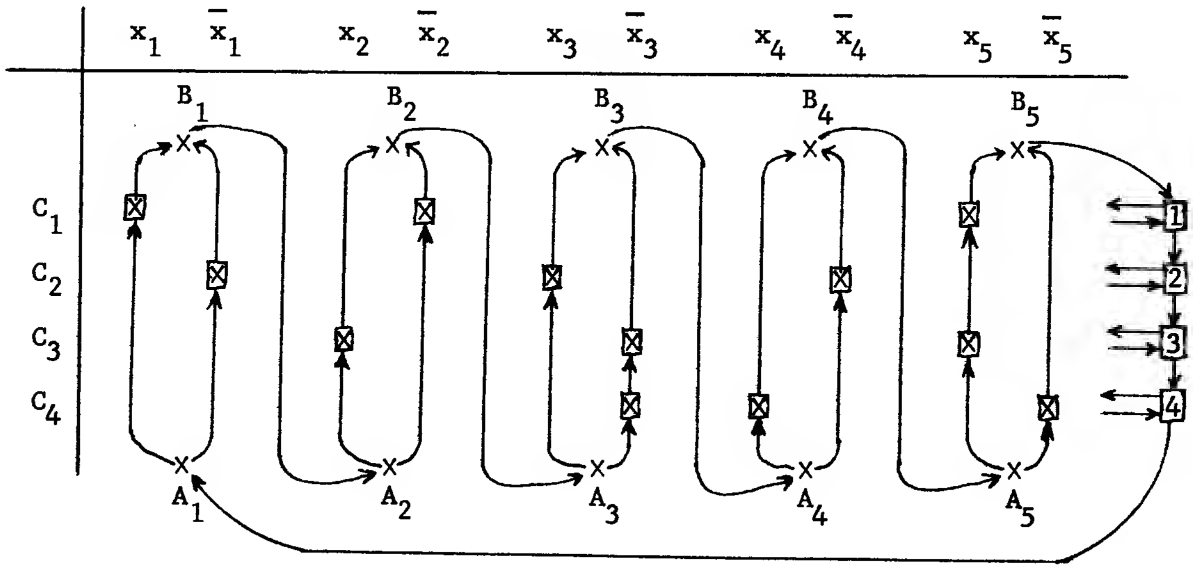
Given a propositional formula F in conjunctive form. Assume F has m clauses C_i, $i = 1, 2, \ldots, m$, and r variables x_j occur in F, $j = 1, 2, \ldots, r$. - Essentially we use the same array as the one used in (3.4). We have a row i for every clause C_i and columns for $x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_r, \bar{x}_r$. We draw a box $\boxtimes$ in a column in Row i iff the literal corresponding to this column occurs in C_i. In addition we draw a cross A_j below and a cross B_j above each double column j, $j = 1, 2, \ldots, r$, and a box $\boxed{i}$ for each row i, $i = 1, 2, \ldots, m$, on the right hand side of the array.

<u>Example</u>: $F = C_1 \wedge C_2 \wedge C_3 \wedge C_4$,

$C_1 = x_1 \vee \bar{x}_2 \vee x_5$, $C_2 = \bar{x}_1 \vee x_3 \vee \bar{x}_4$,

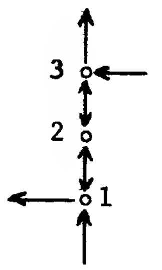
$C_3 = x_2 \vee \bar{x}_3 \vee x_5$, $C_4 = \bar{x}_3 \vee x_4 \vee \bar{x}_5$.

[+]) HC = Hamiltonian Circuit.


x_1 $\bar{x}_1$ x_2 $\bar{x}_2$ x_3 $\bar{x}_3$ x_4 $\bar{x}_4$ x_5 $\bar{x}_5$
B_1 B_2 B_3 B_4 B_5
C_1 C_2 C_3 C_4
1 2 3 4
A_1 A_2 A_3 A_4 A_5

From the array we construct a graph G. The crosses will become nodes, and the boxes $\boxed{\times}$ and $\boxed{i}$ are going to be subgraphs of G.

Each subgraph $\boxed{\times}$ is isomorphic to the following one having three nodes:



There is one chain of arcs upwards along each column of the array, entering the subgraphs $\boxed{\times}$ at 1 and leaving them at 3. For a double column the two chains start at A_j on the bottom and end at B_j on the top. B_j is connected to A_{j+1}, $j = 1, 2, \ldots, r-1$, and B_r to $\boxed{1}$ by arcs.

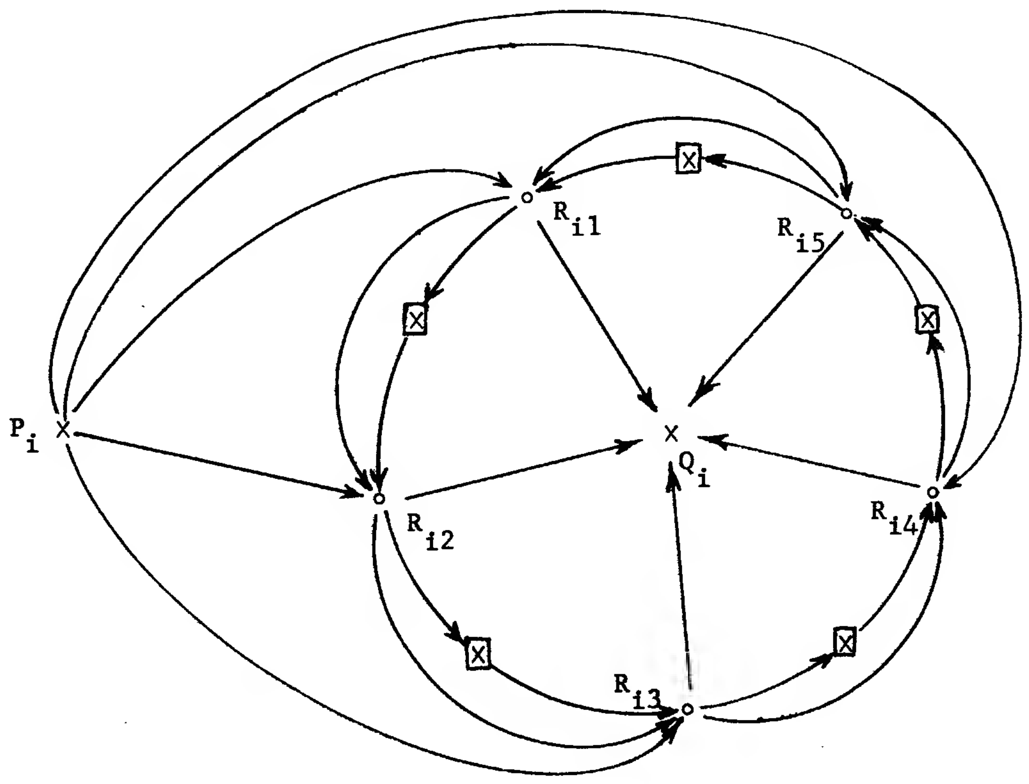
There are arcs from $\boxed{i}$ to the $\boxed{\times}$'s of Row i entering an $\boxed{\times}$ always at its Node 3, and we have an arc from each $\boxed{\times}$ of Row i to $\boxed{i}$ leaving the $\boxed{\times}$ at its Node 1. The $\boxed{i}$'s are described below, but each $\boxed{i}$ has - among other nodes - an entrance node P_i and an exit node Q_i. B_r is connected to P_1, Q_i to P_{i+1} for $i = 1, 2, \ldots, m-1$, and

Q_m to A_1.

The idea of the construction is the following. A HC must go through A_1 and then follow either Column x_1 or Column $\bar{x}_1$ up to B_1. Observe that the structure of the subgraphs $\boxed{X}$ and the way they are connected with each other along the columns forces the path to proceed along one and the same column after A_j is passed until B_j is reached. Depending on the truth assignment, we choose Column x_1 if x_1 is true, and Column $\bar{x}_1$ if x_1 is false. From B_1 we come to A_2 and go up Column x_2 if x_2 is true and up Column $\bar{x}_2$ if x_2 is false, and so on.

After the path has reached B_r it must pick up all the nodes in the $\boxed{i}$'s, starting at $\boxed{1}$, then going to $\boxed{2}$, and so on, until $\boxed{r}$ is reached. While picking up the nodes of $\boxed{i}$, the rest of the nodes in Row i must be picked up. Given there are r_i literals in Clause i, $\boxed{i}$ is constructed in such a way that from it at the most $r_i - 1$ $\boxed{X}$'s can be picked up.

As far as the $\boxed{i}$'s are concerned, our original solution was for three literals per clause only. It has been generalized and substantially improved by Michael J. Fischer and Joel Seiferas. They gave the following elegant solution for what the subgraph $\boxed{i}$ has to perform. For example, assume we have 5 literals in Clause i, i.e. 5 $\boxed{X}$'s in Row i. These $\boxed{X}$'s are connected with $\boxed{i}$ as follows: (See the figure on the next page.)



$\boxed{i}$ has nodes P_i, Q_i, and R_{ij}, j = 1, 2, 3, 4, 5. It is easy to convince oneself that a path from P_i to Q_i can pick up all the R_{ij}'s and an arbitrary proper subset of the $\boxed{X}$'s, but never all the $\boxed{X}$'s. - The generalisation for any number of literals per clause is obvious.

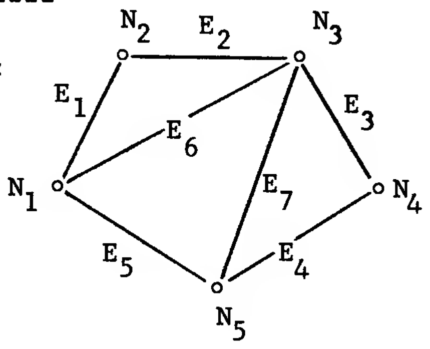
Thus we have shown: F is satisfiable iff G has a HC.

(3.6) Undirected HC $\longleftrightarrow$ SAT.

We describe a very simple reduction requiring only linear space. This reduction is best described by an example.

Example:

Graph G:



G has nodes N_i and edges E_j;

i = 1, 2, 3, 4, 5;

j = 1, 2, 3, 4, 5, 6, 7.

Consider Node N_1. A HC must include

$$E_1 \text{ and } E_5 \text{ and not } E_6$$
$$\text{or} \quad E_1 \text{ and } E_6 \text{ and not } E_5$$
$$\text{or} \quad E_5 \text{ and } E_6 \text{ and not } E_1 \ .$$

Therefore we associate a Boolean variable x_j with Edge E_j, $j = 1, 2, 3, 4, 5, 6, 7$, and write the following five formulas:

F_1: $x_1x_6\bar{x}_5 \vee x_1x_5\bar{x}_6 \vee x_5x_6\bar{x}_1$

F_2: x_1x_2

F_3: $x_2x_3\bar{x}_6\bar{x}_7 \vee x_2x_6\bar{x}_3\bar{x}_7 \vee x_2x_7\bar{x}_3\bar{x}_6 \vee x_3x_6\bar{x}_2\bar{x}_7 \vee x_3x_7\bar{x}_2\bar{x}_6 \vee x_6x_7\bar{x}_2\bar{x}_3$

F_4: x_3x_4

F_5: $x_4x_5\bar{x}_7 \vee x_4x_7\bar{x}_5 \vee x_5x_7\bar{x}_4$.

It is easy to see that

$$F = F_1 \wedge F_2 \wedge F_3 \wedge F_4 \wedge F_5$$

is satisfiable iff G has a HC.

Remark: The reduction can be changed easily to the reduction directed HC $\longmapsto$ SAT .

Acknowledgements

Much of this work is due to the discussions I had with Prof. Albert R. Meyer and Prof. Michael J. Fischer. Their help and interest are greatly appreciated.

References:

[1] Stephen A. Cook, "The Complexity of Theorem-Proving Procedures", Proceedings of Third Annual ACM Symposium on Theory of Computing, 1971.

[2] Stephen A. Cook, "A Hierarchy for Nondeterministic Time Complexity", Proceedings of Fourth Annual ACM Symposium on Theory of Computing, 1972.

[3] J. Edmonds, R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", Report, Operations Research Center, U. of C., Berkeley, July 1970.

[4] M. J. Fischer, A. R. Meyer, M. Paterson, "A Note on Disjunctive Form Tautologies", unpublished paper, 1972.

[5] R. S. Garfinkel, G. L. Nemhauser, "Integer Programming", J. Wiley and Sons, 1972.

[6] Marshall Hall, Jr., "An Algorithm for Distinct Representatives", American Mathematical Monthly, Vol. 63, 1956.

[7] Richard M. Karp, "Reducibility among Combinatorial Problems", in 'Complexity of Computer Computations', R. E. Miller and J. W. Thatcher, ed., Plenum Press, N.Y.

[8] Eugene Lawler, Lecture Notes on Combinatorial Theory, unpublished, 1971/1972.

[9] Albert R. Meyer, Lecture Notes on Algorithms, unpublished, Fall 1972.

[10] Michael Rabin, in "Complexity of Computer Computations" (panel discussion), edited by R.E. Miller and J. W. Thatcher.

[11] Sartaj Sahni, "Some Related Problems from Network Flows, Game Theory and Integer Programming", 13th Annual Symposium on Switching and Automata Theory, 1972.

[12] L. J. Stockmeyer, "Planar 3-Colorability is Polynomial Complete", to appear in SIGACT NEWS.

CS-TR Scanning Project
Document Control Form

Date : 2/23/96

Report # LCS-TR-113

Each of the following should be identified by a checkmark:
Originating Department:

- ☐ Artificial Intellegence Laboratory (AI)
- ☒ Laboratory for Computer Science (LCS)

Document Type:

- ☒ Technical Report (TR)
- ☐ Technical Memo (TM)
- ☐ Other:

Document Information

Number of pages: 32 (38-images)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
- ☒ Single-sided or
- ☐ Double-sided

Intended to be printed as :
- ☐ Single-sided or
- ☒ Double-sided

Print type:

☐ Typewriter	☐ Offset Press	☐ Laser Print
☐ InkJet Printer	☒ Unknown	☐ Other:

Check each if included with document:

☐ DOD Form	☐ Funding Agent Form	☒ Cover Page
☐ Spine	☐ Printers Notes	☐ Photo negatives

☒ Other: BIBLIOGRAPHIC DATA SHEET

Page Data:

Blank Pages (by page number):

Photographs/Tonal Material (by page number):

Other (note description/page number):

Description : Page Number:

IMAGE MAP! (1-32) UN#'ED TITLE PAGE, 2-32
(33-38) SCANCONTROL, COVER, BIBLIO-DATA SHEET, TRGT'S (3),

Scanning Agent Signoff:
Date Received: 2/23/96 Date Scanned: 3/19/96 Date Returned: 3/21/96

Scanning Agent Signature: Michael W. Cook

Rev 9/94 DS/LCS Document Control Form cstrform.vsd